SAS® GLOBAL FORUM 2018

USERS PROGRAM

Running Parts of a SAS Program While Preserving the Entire Program

Stephen Sloan
Accenture

April 8 – 11 | Denver, CO
#SASGF

Running Parts of a SAS Program While Preserving the Entire Program

Stephen Sloan

Accenture

ABSTRACT

The Challenge: We have long programs that accomplish a number of different objectives. We often only want to run parts of the programs while preserving the entire programs for documentation or future use. Some of the reasons for selectively running parts of a program are:

Part of it has run already and the program timed out or encountered an unexpected error. It takes a long time to run so we don't want to re-run the parts that ran successfully.

We don't want to recreate data sets that were already created. This can take a considerable amount of time and resources, and can also occupy additional space while the data sets are being created.

We only need some of the results from the program currently, but we want to preserve the entire program.

We want to test new scenarios that only require subsets of the program.

%INCLUDE statements to call multiple programs from within a shell SAS program

Place each logically distinct module in a separate SAS program.

Have the main program issue a series of %INCLUDE statements to run the entire program.

Comment out the parts you don't need.

Here is an example:

```
%INCLUDE program A;
*%INCLUDE program B;
%INCLUDE program C;
```

Use the operating system to call the different parts of the program

Place each logically distinct module in a separate SAS program.

Have the operating system call each program separately.

Comment out the parts you don't need.

```
sas pgma.sas
#sas pgmb.sas
sas pgmc.sas
```

Running Parts of a SAS Program While Preserving the Entire Program

Stephen Sloan

Accenture

Pass the necessary values to the program

For the two techniques on the previous slide, you will need to ensure that necessary values are passed:

Create text files with the parameters.

Use the command line option -SYSPARM to take in the values. Then read them in the program by using the SYSPARM() function or the &SYSPARM macro variable.

Save the data sets needed later in permanent SAS data sets. These can be deleted at the end of the last program using PROC DELETE or PROC DATASETS. If all of the datasets to be deleted are in the same library, and they are the only members of the library, you can use the KILL function that is part of PROC DATASETS.

Use macros to deactivate code

Put the code that you don't want run inside a macro

Don't execute the macro

This will work even if there are macros inside the macro you create

```
SAS code you want to run
%MACRO SKIP;
SAS code you don't want to run
%MEND SKIP;
SAS code you want to run
```

Use %LET statements to indicate to macros which parts of the program to run

Put the %LET statements at the beginning of the program to make them easier to modify

Set a flag to identify which parts of the program to run

```
* The following statements will run the code *;
%LET FLAGA=Y;
%MACRO RunA(A);
%IF &A=Y %THEN %DO;
                    SAS statements;
%END;
%MEND RunA;
%RUNA(&FLAGA);  *** The value is Y, so the statements will run***;
```

Use macros to deactivate code

Put the code you don't want to run between /* and */

WARNING: If the code already contains */, this will not work because the comment will end too early

```
/**************** Don't run the following code
          SAS code you don't want to run;
*******************/
*Run the following code:;
SAS code you want to run;
```

Running Parts of a SAS Program While Preserving the Entire Program

Stephen Sloan

Accenture

Use SAS EG interactively to only submit the parts of the program you want to run

Open SAS EG
Click on **File**
Click on **Open**
Click on **Program**
Import the program
Highlight the part you want to run
Click on the downward pointing arrow next to Run
Click on **Run Selection**

Add CANCEL to the RUN statement

Add CANCEL to the RUN statement. This will cause the DATA or PROC step not to be run.
There are exceptions where CANCEL will not prevent execution:

- A data step with a DATALINES statement
- PROC SQL

Here is an example of using the CANCEL option:

- PROC FREQ DATA=x;
- TABLES y;
- RUN CANCEL;

Combine more than one method

Combine more than one method

- Set up a shell program to run the %INCLUDEs
- Use %LET to give the paths or the names of the programs to be included

You can combine the above techniques as required and you will have a workable solution. You will be able to keep entire SAS programs intact and only run the parts that are required at the time.

Conclusion

There are a number of methods that allow us to run parts of SAS programs while leaving the integrity of the program intact.

These methods allow us to keep the program together in one spot. They can be used separately or in conjunction with each other.

Suggestions and questions are encouraged and appreciated:

Stephen Sloan
Senior Principal
Accenture
Stephen.b.sloan@Accenture.com
917-375-2937

SAS® GLOBAL FORUM 2018

April 8 – 11 | Denver, CO
Colorado Convention Center

#SASGF

Paper 2511-2018

Running Parts of a Program while Preserving the Entire Program

Stephen B. Sloan, Accenture

The Challenge: We have long programs that accomplish a number of different objectives. We often only want to run parts of the programs while preserving the entire programs for documentation or future use. Some of the reasons for selectively running parts of a program are:

- Part of it has run already and the program timed out or encountered an unexpected error. It takes a long time to run so we don't want to re-run the parts that ran successfully.
- We don't want to recreate data sets that were already created. This can take a considerable amount of time and resources, and can also occupy additional space while the data sets are being created.
- We only need some of the results from the program currently, but we want to preserve the entire program.
- We want to test new scenarios that only require subsets of the program.

Possible solutions:

%INCLUDE statements

Place each logically distinct module in a separate SAS program.
Have the main program issue a series of %INCLUDE statements to run the entire program.
Comment out the parts you don't need.
Here is an example:

```
%INCLUDE program A;
*%INCLUDE program B;
%INCLUDE program C;
```

Use the operating system to call the different parts of the program

Place each logically distinct module in a separate SAS program.
Have the operating system call each program separately.
Comment out the parts you don't need. This is similar to the above method but it is done in a Unix Script

```
sas pgma.sas
#sas pgmb.sas
sas pgmc.sas
```

For the above two techniques, you will need to do the following to ensure that necessary values are passed:

This requires planning to ensure that the necessary macro variables and work data sets are passed to the succeeding programs. This can be done in the following ways:

Create text files with the parameters being passed.

Use the command line option -SYSPARM to take in the values when using the operating system commands. Then read them in the program by using the SYSPARM() function or the &SYSPARM macro variable.

Save the data sets needed later in permanent SAS data sets. These can be deleted at the end of the last program using PROC DELETE or PROC DATASETS. If all of the datasets to be deleted are in the same library, and they are the only members of the library, you can use the KILL function that is part of PROC DATASETS.

Use macros to deactivate code

Put the code that you don't want run inside a macro
Don't execute the macro
This will work even if there are macros inside the macro you create

```
SAS code you want to run
%MACRO SKIP;
SAS code you don't want to run
%MEND SKIP;
SAS code you want to run
```

Use %LET statements to indicate to macros which parts of the program to run

Put the %LET statements at the beginning of the program to make them easier to modify
Set a flag to identify which parts of the program to run

```
*** The following statements will run the first set of code but not the second ***;
%LET FLAGA=Y;
%LET FLAGB=N;

%MACRO RunA(A);
        %IF &A=Y %THEN %DO;
        SAS statements;
        %END;
%MEND RunA;
%RUNA(&FLAGA);  *** The value is Y, so the statements will run***;

%MACRO RunB(B);
        %IF &B=Y %THEN %DO;
        SAS statements;
        %END;
%MEND RunB;
%RUNB(&FLAGB);   *** The value is N so the statements will not run ***;
```

Comment out parts of the program

Put the code you don't want to run between /* and */
WARNING: If the code already contains */, this will not work because the comment will end too early.
In this case you will need to comment out the parts before and after the comments with /*

```
/**************** Don't run the following code
SAS code you don't want to run
*******************/
*Run the following code:
SAS code you want to run
```

Using SAS EG interactively to only submit the parts of the program you want to run

Open SAS EG
Click on **File**
Click on **Open**
Click on **Program**
Import the program
Highlight the part you want to run
Click on the downward pointing arrow next to Run
Click on **Run Selection**

Add cancel to the RUN statement

Add CANCEL to the RUN statement. This will cause the DATA or PROC step not to be run.
There are exceptions where CANCEL will not prevent execution:

A data step with a DATALINES statement
PROC SQL

Here is an example of using the CANCEL option:

```
PROC FREQ DATA=x;
      TABLES y;
RUN CANCEL;
```

Combine more than one method

Set up a shell program to run the %INCLUDEs
Use %LET to give the paths or the names of the programs to be included

You can combine the above techniques as required and you will have a workable solution. You will be able to keep entire SAS programs intact and only run the parts that are required at the time.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Email: stephen.b.sloan@accenture.com